>: FAQs//

Content Processing Development Kit

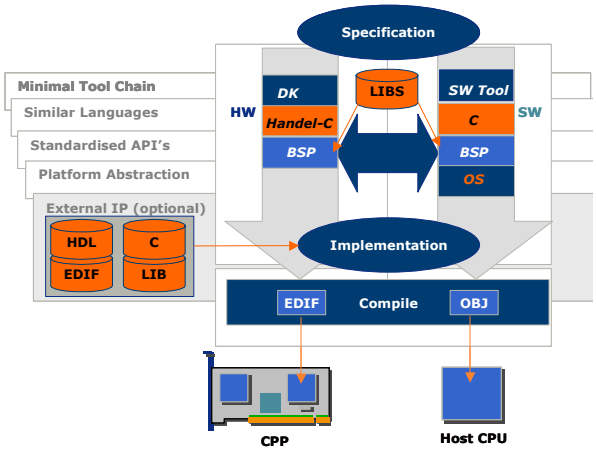

Fig 1. *Development Flow*

Q1. What is the Content Processing Development Kit (CP-DK)?

Q2. What does the Content Processing Development Kit (CP-DK) include?

Q3. What is the Content Processing Platform (CPP)?

Q4. How does a CPP help me to accelerate my software applications?

Q5. How can I design (HW) acceleration IP?

Q6. Why should I consider using CP-DK?

Q7. How does Celoxica's DK design suite help me to develop content for the CPP platform?

Q8. What specific features are available in CP-DK to develop optimized designs?

Q9. Why should I consider Handel-C for hardware development?

Q10. What is the relationship between ANSI-C and Handel-C?

Q11. What are the key benefits of making direct calls between Handel-C and C/C++?

Q12. Given that C is a sequential language, how does Handel-C address hardware specific issues e.g. implementation of parallelism?

Q13. What is your recommended development flow for Windows OS server based application acceleration? Linux based servers and clusters?

Q14. How do I evaluate the tradeoffs of executing my algorithm on the CPU versus the CPP?

Q15. What are the minimum development system requirements for developing (HW) acceleration IP using CP-DK?

Q16. Does Celoxica have design examples and tutorials for the CPP and how do I obtain them?

Q17. What are the other Content Processing building blocks from Tarari?

Q18. What are the other Content Processing acceleration products from Tarari?

Q19. What are the main differences between this approach and the industry's standard approach to address content processing platforms?

Q20. What are the main advantages of an FPGA-based platform versus an ASIC one?

Q21. Where can I get more information about CP-DK and the Content Processor Platform?

Q22. How can I schedule a presentation on CP-DK?

Q23. How can I purchase CP-DK?

Q24. What else do I need to purchase to develop Content Processors?

Q25. How can I get support for CP-DK?

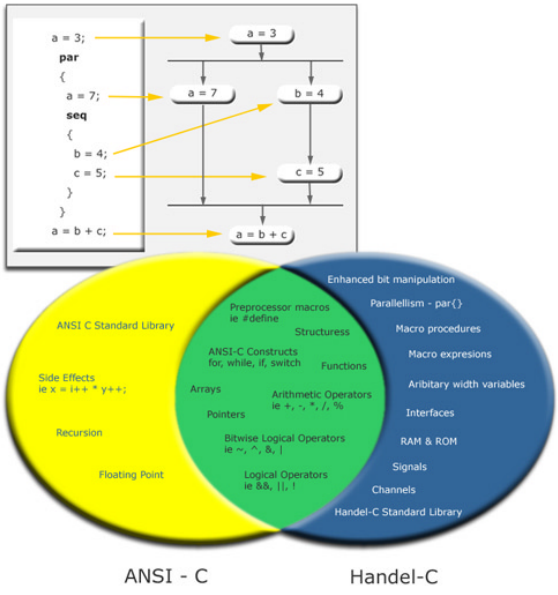

Fig 2. *ANSI-C/ Handel-C comparison*

>: FAQs//

Content Processing Development Kit

Q1. What is the Content Processing Development Kit (CP-DK)?

A1. CP-DK is a combination of the hardware and software content processing building blocks that creates a flexible, dynamically reconfigurable hardware platform designed to accelerate a variety of compute-intensive algorithms. CP-DK also includes development tools, sample code and design and usage guides that enables developers to quickly ramp their learning curve.

Q2. What does the Content Processing Development Kit (CP-DK) include?

A2. CP-DK includes the hardware to create a Content Processor, namely a Content Processing Platform (CPP), currently a ½ length PCI card. It also includes the software components, drivers libraries, development aids and documentation required by designers to develop Content Processors that snap into servers, appliances and network devices.

Q3. What is the Content Processing Platform (CPP)?

A3. A PCI card incorporating reconfigurable logic that can target specific compute-intensive tasks and decrease the processing time required to perform specific operations.

The CPP (Figure 3) consists of

- Two CPE's – Content Processing Engines, each is a *Virtex-II 1000 FPGA* with connections to:
  - Eight LEDs
  - 2x 1MB SRAM
  - CPC
- CPC – Content Processing Controller, with connections to:
  - 256MB DDR SDRAM
  - PCI Bus to Host

Q4. How does a CPP help me to accelerate my software applications?

A4. There are two key opportunities with a hardware co-processor:

- **Algorithm Acceleration**: Exploit the parallelism in algorithms to increase performance with implementation in custom (parallel) hardware i.e. FPGA's.
- **Algorithm Offload**: Exploit the coprocessor to free CPU resource

For PCI-based coprocessor cards candidate algorithms include ones where CPU execution time is comparable to or far exceeds data transfer time over PCI.

A full analysis needs to consider:

- Time required to perform the algorithm in the co-processor.
- System application performance improvement – Amdahl's Law.

Q5. How can I design (HW) acceleration IP?

A5. There are two approaches. We recommend and use the second.

- Traditional Options – HDL based design
  - Purchase FPGA (HW) development tools
  - Hire/use HW engineers
  - Pay 3rd Party development fees
- The Alternative – "Software Compiled System Design"
  - Use Celoxica **Content Processing Development Kit**
  - Development framework with Example Acceleration IP
  - Comprehensive Hardware-Software Co-simulation environment
  - Tool and Language Connectivity
  - Enable SW engineers and/or increase HW engineer productivity

Further benefits of CP-DK are outlined below.

Q6 Why should I consider using CP-DK?

A6. The development flow (Figure 1) using CP-DK provides:

- **A minimal tool chain:** Celoxica DK design suite, C/C++ compiler and Xilinx Place and Route tools.
- **A common language base:** – mixed language descriptions, C and Handel-C.
- **API's for common interfacing and platform abstraction:** Simplifying application development

CP-DK:

- **Reduces the time taken to design acceleration IP**

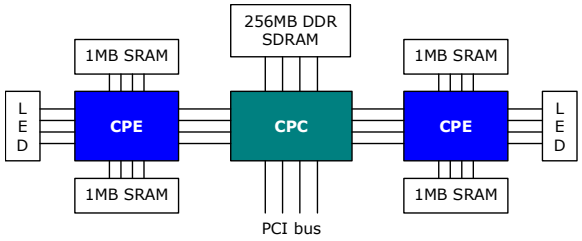

Fig 3. *Basic CPP Architecture.*

  - Reduces cost, time to market and risk
  - SW-HW Partition exploration using system models for better designs faster
- **Widens access to the design of acceleration IP**
  - Closer cooperation between the hardware and software designers
  - Better utilisation of existing resources
- **Full integration with other tools**
  - C/C++, RTL, ISS, Block Model Simulators (MatLab)
  - Automation of FPGA configuration tools Enables complete system design using the most appropriate technology

Q7. How does Celoxica's DK design suite help me to develop content for the CPP platform?

A7. DK (Figure 6) is Celoxica's integrated design environment (IDE) allows the user to take C language level code (Handel-C) directly to hardware.

DK provides a software development-like environment that enables designers to work with a common methodology in both hardware and software implementation. It provides facilities to:

- Manage and edit multiple source files.
- Simulate and debug complex designs.
- Co-simulate with C and C++ language source code.
- Co-simulate with HDL implementations and other development environments
- Integrate and build hardware.

Q8. What Specific features are available in CP-DK to develop optimised designs?

A8. The similar HW/SW language and methodology (Figure 1) simplify the exploration of system partitions and allocation of functionality between hardware and software for optimum performance. Designing applications to a well-defined API allows for fast simulations (Figure 4) and many design iterations to be tested against performance requirements.

Logic area and timing estimation output is presented in an HTML format report. Each line of code has an associated report of the number of LUTs, flip-flops, memory bits and other additional logic elements.

Information on the longest path in a design is also presented to aid the designer in accurate analysis of critical path timing and circuit behavior.

This allows the designer to experiment with different optimization strategies within the DK environment until their design goals are met, before placement of the design.

Q9. Why should I consider Handel-C for hardware development?

A9. CPP based systems combine both hardware & software. Consequently, they emphasize the need to manage software as well as hardware components through effective *co-design*. When coupled with increasingly complex algorithmic designs this means that the decision to move to a language with a higher level of abstraction is a logical one. This is the hardware design analogue of using C and a C-language compiler as opposed to programming a software system in assembler.

For an efficient and timely co-design solution it is advantageous to use a common language base for both the modeling and implementation of the overall system. This will facilitate its effective planning, partitioning and implementation and with similar design methods improves the optimal allocation of functionality to both the hardware and

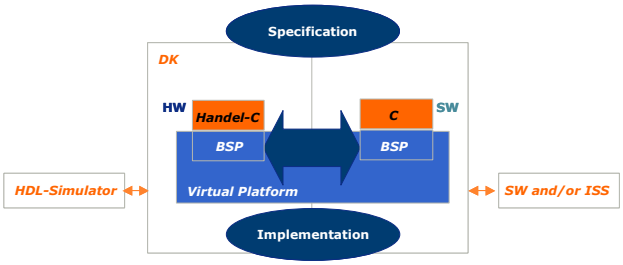

Fig 4. *API's enable rapid co-verification.*

software resources.

Handel-C represents the next logical step in the evolution of hardware design, just as HDLs were a step above Schematic Capture. Handel-C enables design at the functional, rather than the structural level, making it suitable for prototyping and implementing algorithms in hardware, hardware/ software co-design and design space exploration.

Q10. What is the relationship between ANSI-C and Handel-C?

A10. Based on the structure, syntax and language elements of ANSI-C, Handel-C has extensions required for hardware development (Figures 2 and 5). These include:

- Parallel processing.
- Flexible data widths.
- Multiple clock domains.
- Communications between parallel elements.
- A simple timing model.

Q11. What are the key benefits of making direct calls between Handel-C and C/C++?

A11. This capability enables designers to:

- Create system models in C/C++.
- High level verification of the functionality and characteristics of a system.
- Select individual modules for hardware implementation in Handel-C.
- Make more informed partitioning decisions and to make optimum use of the processor plus hardware (acceleration card) resources.
- Simulate Handel-C hardware modules within the overall system, acting as a '*test bench'* to verify Handel-C functionality.

Q12. Given that C is a sequential language how does Handel-C address hardware specific issues e.g. implementation of parallelism?

A12. A language construct, the 'par' statement has been implemented in Handel-C to allow the designer to specify code that should execute in parallel (by building parallel blocks of logic) thereby giving control over the performance (and size) of the design.

In addition Handel-C uses a powerful, yet simple, timing model where each assignment in the program takes one clock cycle to execute, with specific types for sub-clock cycle operations, giving the designer full control over what is happening in the design at any point in time.

E.g. these three assignments execute in parallel and in the same clock cycle:

```
par
{
     x = 1;
     y = 2;
     z = 3;}
```

```
define WIDTH 9       // parameterisable data widths

typedef struct       // complex number type
{
    signed WIDTH re;
    signed WIDTH im;
}
complex;

set clock = external " ClockSource ";
void main()          // single synchronous clock domain (ClockSource)
{
    chan complex cDataIn[2], cDataOut;        // communication channels

    while(1)
          par        // parallel hardware
          {
               DataIO(cDataIn, &cDataOut);              // data input/output function
               Transform(cDataIn, &cDataOut);           // data transform function
          }
}

void Transform(chan complex *pcDataIn, chan <complex> *pcDataOut)
{
    complex DataInReg[2],DataOutReg;         // complex data registers

   par (i=0; i<2; i++)               // replicted par{}
    {
          pcDataIn[i] ? DataInReg[i];       // read complex numbers in parallel
    }
    par
    {   // single cycle multiplication of two complex numbers
          DataOutReg.re = DataInReg[0].re*DataInReg[1].re - DataInReg[0].im*DataInReg[1].im;
          DataOutReg.im = DataInReg[0].re*DataInReg[1].im + DataInReg[0].im*DataInReg[1].re;
    }
    *pcDataOut ! DataOutReg;    // write complex number
}

void DataIO(chan complex *pcDataIn, chan <complex> *pcDataOut)
{
    complex DataInReg[2],DataOutReg;         // complex data registers

    DataInput( &DataInReg[0] );              // data input function
    DataInput( &DataInReg[1] );              // data input function
   par (i=0; i<2; i++)                         // replicted par{}
    {
          pcDataIn[i] ! DataInReg[i]; // write complex numbers in parallel
    }
    *pcDataOut ? DataOutReg;                 // read complex result
    DataOutput(&DataOutReg);                 // data output function
}

#ifdef SIMULATE // Simulation Testbenches

void DataInput(complex *pDataIn)
{
    long Data[2];
    scanf("%d%d", &Data[0], &Data[1]);  // ANSI-C function
    pDataIn->re = adjs(Data[0], WIDTH); // type conversion from ANSI-C call
    pDataIn->im = adjs(Data[1], WIDTH); // type conversion from ANSI-C call
}

void DataOutput(complex *pDataOut)
{
    long Data[2];
    Data[0] = adjs(pDataOut->re, 32);                       // type conversion for ANSI-C call
    Data[1] = adjs(pDataOut->im, 32);                       // type conversion for ANSI-C call
    printf("DataOut: %d %d\n", Data[0], Data[1]);           // ANSI-C function
}

#else // Hardware Implementations

void DataInput(complex *pDataIn)
{
    interface port_in(signed WIDTH in) DataInPort();  // WIDTH-bit input port
    pDataIn->re = DataInPort.in;
    pDataIn->im = DataInPort.in;
}

void DataOutput(complex *pDataOut)
{
    signed WIDTH Data;
    interface port_out() DataOutPort(signed WIDTH OutPort = Data); // WIDTH-bit output port
    Data = pDataOut->re;
    Data = pDataOut->im;
}
#endif
```

Fig 5. *Handel-C code for single cycle multiplication of two complex numbers. For illustration the code deliberately uses Handel-C concepts and extensions such as, par, chan, synchronization, functions, pointers, structures, interfacing and externing pure C functions for simulation.*

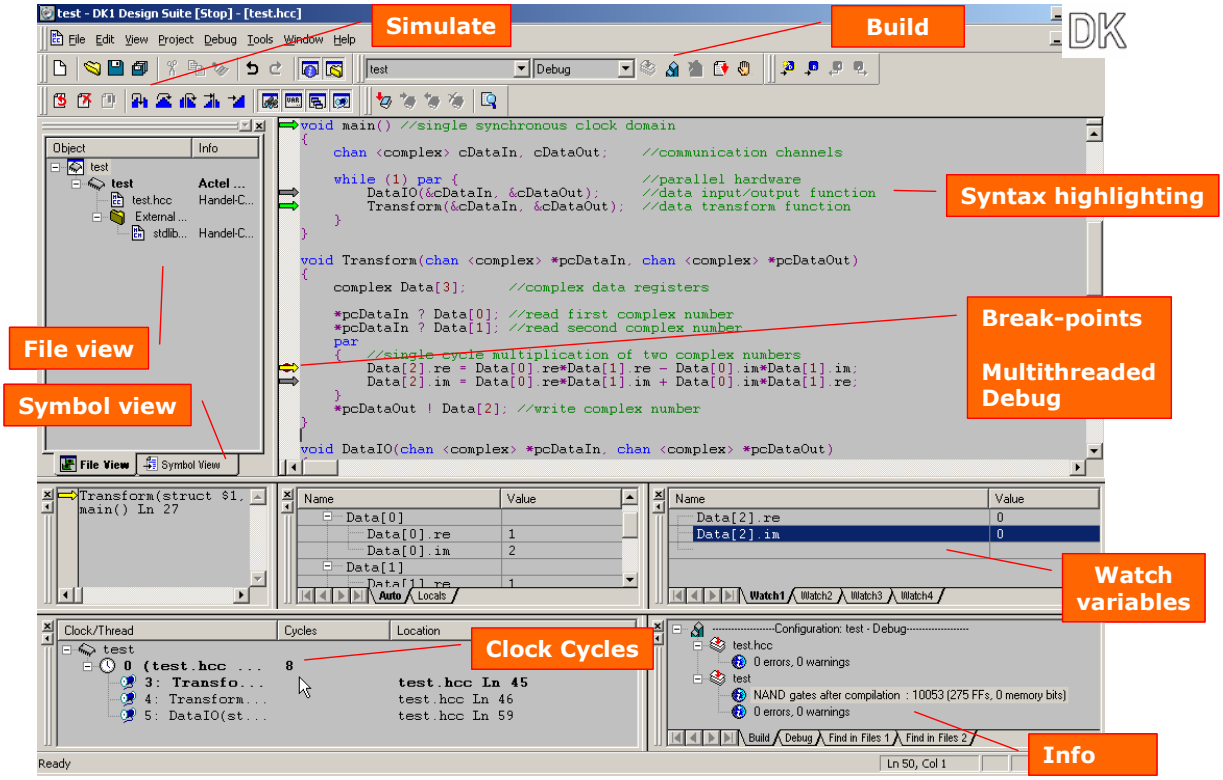

Fig 6. *The DK IDE illustrating parallel symbolic debug.*

The designer can also manage the creation and use of functions within the code, allowing size and performance to be balanced.

Q13. What is your recommended development flow for Windows OS server based application acceleration? Linux based servers and clusters?

A13. The development flow is illustrated in Figures 1 and 4. Development of the HW acceleration IP requires a Windows PC. We suggest:

1. Profile your software application on the target platform (Windows/Linux server) and identify candidate algorithms for implementation on the CPP.
2. Port the core of your application to a Windows PC and create a system model in C/C++.
3. Port or re-implement the candidate portions of the model in Handel-C using the facilities in DK, such as connecting to external C/C++ code, to use the system model as a testbench.
4. Use DK to create a configuration file for the FPGA on the CPP.
5. Port software components back to the target system and test.

Q14. How do I evaluate the tradeoffs of executing my algorithm on the CPU versus the CPP?

A14. Use software profiling to characterize the performance of your application. Using CP-DK you can develop a mixed C/C++/Handel-C system model. DK gives size and speed metrics for the Handel-C portions of the design.

Q15. What are the minimum development system requirements for developing (HW) acceleration IP using CP-DK?

A15. Development of FPGA configuration files requires a windows PC for development. Minimum system requirements for DK are given in the DK

FAQ. Currently only Linux drivers have been released for software development and deployment of CPP platforms. Compatible platforms are given in the Tarari CP-DK fact sheet and product overview.

Q16. Does Celoxica have design examples and tutorials for the CPP and how do I obtain them?

A16. Yes, design examples, application notes and presentations are available from your local FAE or Account Manager or via the support pages at www.celoxica.com.

Q17. What are the other Content Processing building blocks from Tarari?

A17. Tremendous investment has been made to accelerate and process packets as they move through networks and servers. However, this packet processing has been limited to looking at headers, doing table-look up's, and sending the packets on their way. Tarari Content Processors are hardware based subsystem building blocks (silicon, boards, etc.) that snap into servers, appliances and network devices to allow control and inspection of complete messages and rich data at much greater rates than previously possible. Tarari Content Processors ensure that the information in the payloads of these messages can be intelligently accessed and processed while maintaining network speeds.

Q18. What are the other Content Processing acceleration products from Tarari?

A18. CP-DK includes the first of Tarari's Formula-CP Acceleration series of products. Additional acceleration products will be provided in early '03 focused on two main areas: Network Security and XML Web Services.

Q19. What are the main differences between this approach and the industry's standard approach to address content processing platforms?

A19. The focus here is on accelerated processing at the application layer, providing the ability to intelligently analyze the entire payload of the message. The majority of the industry is focused on accelerating network processing at the packet level, looking at only part of the message.

Q20. What are the main advantages of an FPGA-based platform versus an ASIC one?

A20. These include providing the performance and flexibility that leads to a lower cost of ownership. Flexibility is the key that allows Content Processors to adapt as the environment and/or usage pattern changes. The end result is an improved solution that costs less over the lifetime use. In addition, using the Celoxica design methodology can significantly reduce the development times to get a solution up and running enabling faster time-to-market

Q21. Where can I get more information about CP-DK and the Content Processor Platform?

A21. For more information on CP-DK, Celoxica or Tarari visit: www.Celoxica.com/partner/reconfig_apps_default.asp, www.celoxica.com, www.tarari.com or contact your local Celoxica office. You can also email sales@celoxica.com.

Q22. How can I schedule a presentation on CP-DK?

A22. To schedule a presentation you can call Celoxica or Tarari – see contact information at www.celoxica.com and www.tarari.com or send an email to sales@celoxica.com.

Q23. How can I purchase CP-DK?

A23. To purchase CP-DK contact Celoxica directly.

Q24. What else do I need to purchase to develop Content Processors?

A24. To develop Content Processors with the CP-DK libraries and hardware you will also need to purchase the Celoxica DK Design Suite and Xilinx Place and Route Tools. These can be arranged directly with Celoxica.

Q25. How can I get support for CP-DK?

A25. Celoxica is providing support for CP-DK. Send a email to support@celoxica.com or call on the

appropriate telephone number for your region. See www.celoxica.com/support/default.asp.

Customer Support at Support@celoxica.com and +44 (0)1344 663649.

Celoxica Ltd.
20 Park Gate
Milton Park
Abingdon
Oxfordshire OX14 4SH
United Kingdom
Tel: +44 (0) 1235 863 656
Fax: +44 (0) 1235 863 648

Celoxica, Inc
900 East Hamilton Avenue
Campbell, CA 95008
USA
Tel: +1 800 570 7004
Tel: +1 408 626 9070
Fax: +1 408 626 9079

Celoxica Japan KK
YBP West Tower 11F
134 Godo-cho, Hodogaya-ku
Yokohama 240-0005
Japan
Tel: +81 (0) 45 331 0218
Fax: +81 (0) 45 331 0433

Celoxica Pte Ltd
Unit #05-03
31 Int'l Business Park
Singapore
609921
Tel: (65) 6896 4838
Fax: (65) 6566 9213

www.celoxica.com